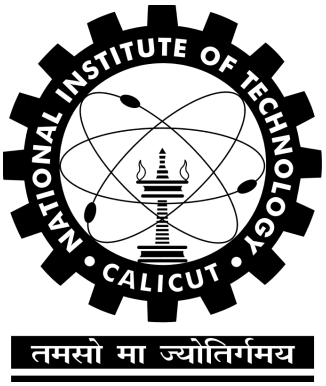
# CS4028D Quantum Computation
# Group Assignment
# Topic - Amazon Braket




**Department of Computer Science and Engineering**
**National Institute of Technology Calicut**

Submitted By: Group 6

Group Name: Quantumplators

GitHub Repo: Getting Started with Amazon Braket

Group Members:


- Amal Francis V Ukken - B180445CS
- Mathew Jose Mammootil - B180586CS
- Meenakshi Madhu - B180390CS
- Neelima Sajeev - B180632CS
- Sidharth Menon - B180561CS
- Vishnu Sajith - B180474CS

# TABLE OF CONTENTS

# Introduction

Quantum computing is a rapidly emerging technology that uses quantum phenomena, including interference, superposition, and entanglement, to solve problems that are too complex for classical computers. Using principles of quantum mechanics could help take a giant leap forward in the space of computation in order to tackle certain complex challenges that are otherwise not feasible. Quantum computers are the machines that perform quantum computing, and they are beneficial for solving and computing complex algorithms, outperforming even the most powerful supercomputers.

Classical computers, which include daily devices such as computers and smartphones, store data in binary "bits" that can be 0s or 1s. On the other hand, A quantum computer uses a quantum bit, or qubit, as its basic memory unit. Physical systems, such as the electron spin or the photon direction, are used to create qubits. These qubits have a property called superposition, which allows them to be in several configurations simultaneously. Quantum computers are built on this foundation.

## Cloud-based quantum computing

The use of quantum emulators, simulators, or processors via the cloud is characterized as cloud-based quantum computing. Since it is practically impossible for users to own quantum computers, cloud services are considered a means of offering access to quantum processing. The first company to start cloud-based quantum computing was IBM in the year 2016. This service helped the users to avail themselves of the facility to create and run small programs on a quantum computer online. Since then, many other tech giants have also started providing such services. Google, Microsoft, Amazon are some of the giants who have started providing cloud services. In 2019, Microsoft unraveled a service called ‘Azure Quantum,’ which allows access to quantum algorithms, hardware, and software. Amazon launched its services known as Amazon Braket in December of 2019. It is a fully managed Amazon Web Services (AWS) cloud service designed to provide quantum computer users remote access to a single development environment.

# AWS Cloud Services

Amazon Web Services (AWS), launched in 2006, is one of the most comprehensive and broadly adopted cloud platforms. It now offers more than 200 fully-featured services, which include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Amazon's highly evolving cloud computing platform can offer organizational tools that include computational power, infrastructure management, database storage, and content delivery services. AWS is marketed by Amazon as a way of obtaining large-scale computational power and resources more quickly and cheaply than building their own servers.

As mentioned above, the Amazon Web Services portfolio comprises more than 200 services, and each can be configured differently based on the user's needs. Some of these services include EC2, lambda (computing), dynamoDB (database), AMS (infrastructure management), Amplify (application development), and IAM (security). For each service, the configuration options and individual server maps are visible to the users.

Amazon Braket, one of the many such services offered by AWS, is a quantum computing service.

# Amazon Braket

Amazon offers a quantum computing service called Amazon Braket. It provides the user with an environment to design, simulate and run their algorithms on different quantum hardware technologies available in the platform. The platform offers a gate-based superconductor quantum computer from Rigetti, a quantum annealing superconductor from D-Wave, and an ion trap quantum computer from IonQ. Building hybrid algorithms that combine classical and quantum tasks is more straightforward with Amazon Braket as it helps manage classical compute resources and establish low-latency connections to the quantum hardware.

The platform is also integrated with other AWS services, thereby enabling monitoring workloads, generating notifications on task completion, and managing access control and permission with the help of Amazon CloudTrail, Amazon EventBridge, and AWS IAM. The results of the simulation are stored in the user-preferred Amazon S3 bucket.

Amazon Braket is currently available in the US East (N. Virginia), US West (N. California), and US West (Oregon) AWS Regions, with more regions planned for the future.

## Features

The platform works with different types of quantum computers and circuit simulators using a consistent set of technology-agnostic development tools. This technology-agnostic SDK removes the need to code against quantum programming environments that commonly exist for different quantum computers. Hence it can be run on any compatible quantum hardware provided through the Amazon Braket Service. The existing designs and algorithms can be tested on new systems if and when they are added to the platform.

They provide a unified environment with a pay-as-you-go pricing model. So the customer has the added benefit of paying only for the computational resources that they use instead of subscribing to the service for a certain period, which is often the case with alternative simulation services on the web.

Amazon Braket allows customers to use the service easily by providing very commonly used tools like Jupyter notebooks to access libraries that help visualize results and collaborate with others. Various pre-built algorithms and tutorials also make the implementation process much smoother for the user.

Rather than building its own quantum hardware, Amazon makes quantum computers from other companies available to cloud users through AWS. Amazon Braket currently supports three different quantum computing services: gate-based superconductor computers from Rigetti, quantum annealing superconductor computers from D-Wave, and gate-based ion trap computers from IonQ. Users can then run their algorithm on any of the available quantum processors without working with multiple tools or committing to a single simulator.

In addition to running quantum algorithms, customers will also be able to use Amazon Braket to run hybrid algorithms, i.e., combining classical computing with quantum computing systems overcoming the limitations present in various quantum computation techniques today. Hybrid quantum algorithms work with an iterative approach, where the quantum computers act as co-processors to classical computing resources.
Amazon Braket supports an open-source software framework called Pennylane, built around the concept of differentiable programming. Users can use this tool to train quantum circuits in the same way as training a neural network. It has an interface similar to that of PyTorch and Tensorflow.

# Facilities

1. **Major Applications**

    - Research - For research purposes in quantum computation, the process is much easier with different tools such as AWS Cloud Credit for Research Program.
    - Hardware research - Braket provides easy access to trapped ion, superconducting, and annealing devices for the process of quantum hardware research.
    - Building quantum software faster - When creating new software for quantum computing, users can quickly bring it into the market using Amazon Braket's software development kit (SDK) and simple pricing and management.

- Exploring industry applications - With the constant advancements in quantum computation, the issue of adapting to the changes arises. With the Amazon Braket, users can prepare their businesses for these advances and learn to optimize, simulate and solve other complex computational problems using the technology.

**2. Different Available Quantum processors**

The quantum computing services provided by Amazon Braket are based on quantum annealing and gate-based quantum computers.

**Quantum Annealers :**

Quantum annealing harnesses quantum mechanical effects like superposition, entanglement, and tunneling to find low-energy configurations that encode the optimal or near-optimal solutions of a problem. The QPUs consist of a network of interconnected superconducting flux qubits, each made from a tiny loop of metal interrupted by a Josephson junction. The loops become superconductors and exhibit quantum-mechanical effects at low temperatures. For a multi-qubit QPU, the qubits are interconnected via superconducting loops called couplers.

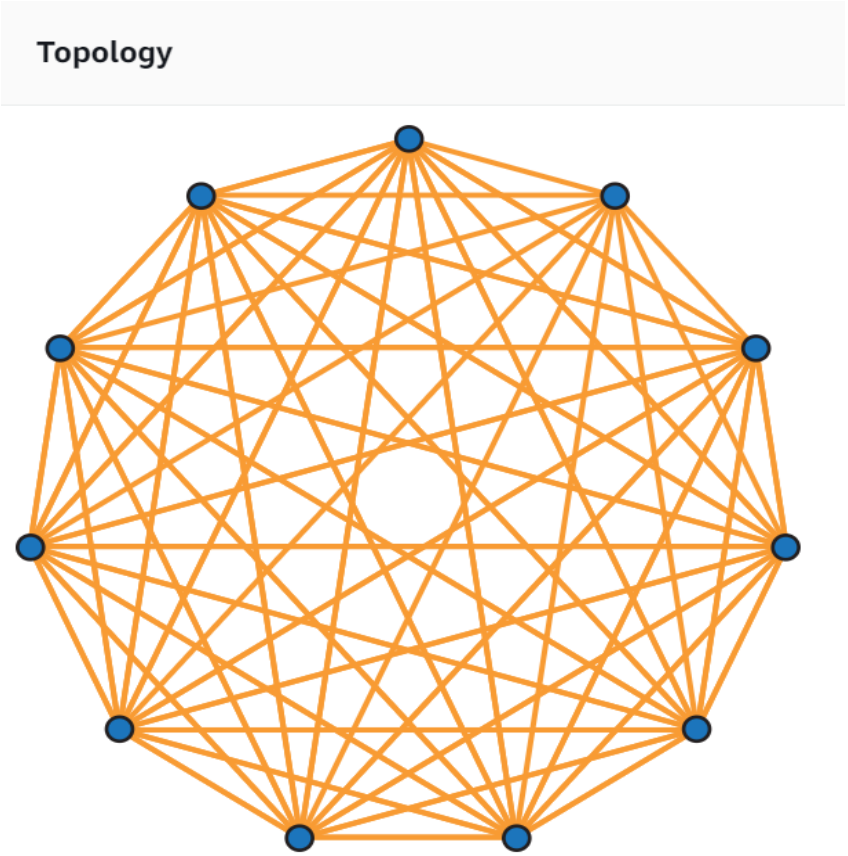
Adding control circuitry to manage the magnetic fields creates an integrated system of programmable quantum devices. Amazon Braket provides access to quantum annealing technology based on superconducting qubits from D-Wave, like the D-Wave 200Q QPU with 2048 qubits and 6000 couplers. 128,000 Josephson junctions are used to reach this scale, which is by far the most complex superconducting integrating circuit ever built. The topology of the qubits is in an architecture known as Chimera.

**Gate-based ion-trap processors :**

Trapped ion quantum computers implement qubits using electronic states of charged atoms (ions) in a cryogenic vacuum. A linear ion trap is a specialized chip featuring around 100 tiny electrodes precisely designed and lithographed to produce the electromagnetic forces that hold the ion in 3D space. The ions are isolated from the environment to minimize environmental noise and decoherence. The quantum-mechanical state defines the qubit: the ground state is defined as a logical 1, and the excited state is a logical 0. After gate operations are performed, the resulting qubit state is read using a readout laser.

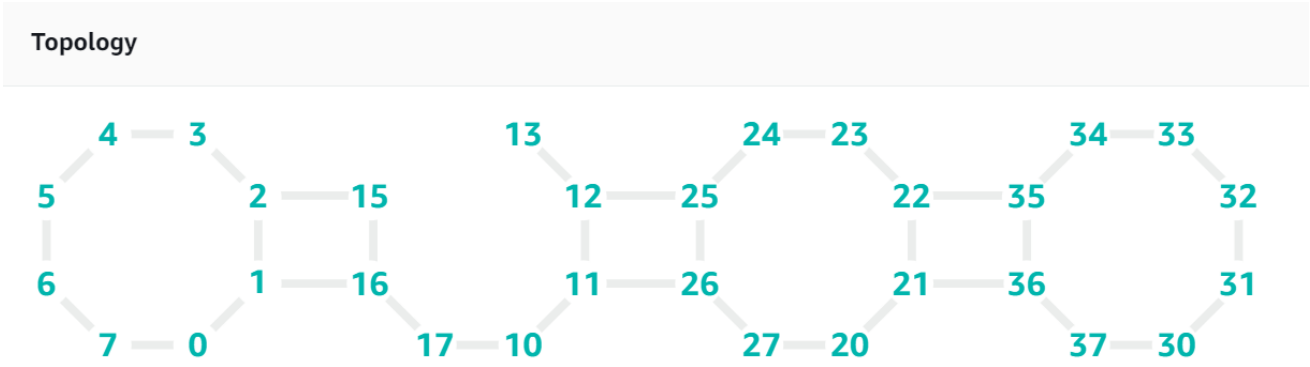
Amazon Braket provides access to ion trap quantum computers from IonQ.

IonQ is a universal gate quantum computer with 11 qubits. These qubits are made out of Ytterbium and trapped with lasers in a chain. This device is fully connected, i.e., each qubit has a direct connection to every other qubit. This becomes very important for efficient quantum algorithms.




**Gate-based superconducting processors :**
Superconducting qubits are built with superconducting electric circuits operating at cryogenic temperatures. Qubits are coupled to a linear superconducting resonator for readout. The combination of the qubit, the linear readout resonator, and the associated wiring provides a general-purpose quantum circuit element capable of reliably encoding, manipulating, and reading out quantum information. Rigetti processors use arrays of qubits coupled to one another with on-chip capacitances. Microwave or DC pulses are used to implement single and multi-qubit logic operations.

Amazon Braket provides access to quantum hardware based on superconducting qubits from Rigetti, like the Rigetti Aspen-8. This device is a universal gate-based quantum computer with 30 superconducting qubits. The qubits here are not directly connected.




### 3. Simulators available on AWS

Amazon provides us with four different quantum circuit simulators - one local simulator and three fully managed simulators. The local simulator is included in the Braket SDK and is free of cost with up to 25 qubits. The fully managed simulators are SV1, TN1, and DM1. SV1 provides up to 34 qubits and costs around 0.075 USD per minute. DM1 is designed for noise modeling and costs the same as SV1. TN1 offers up to 50 qubits and costs 0.275 USD per minute.

### 4. Training Provided By AWS

Amazon provides us with different tools and services for us to get started with Amazon Braket. The Amazon Braket Jupyter notebooks have a selection of algorithms and models that have been built, tested, and could be used by customers. AWS also provides training functionality for customers to get comfortable with Amazon Braket through Quantum Solutions Lab.

# Amazon Braket - Drawbacks

Amazon Braket, even though being a major player in the quantum computing field, still has some drawbacks. Compared to its biggest competitor IBM Q, Braket doesn't offer a free service. In comparison, IBM Q provides both free and premium services to its users. In Braket, the user is billed separately for each Amazon Braket capability, including access to quantum computing hardware and managed simulators. The customers will also be charged separately for the AWS services provided through Amazon Braket, such as Amazon Braket managed notebooks.

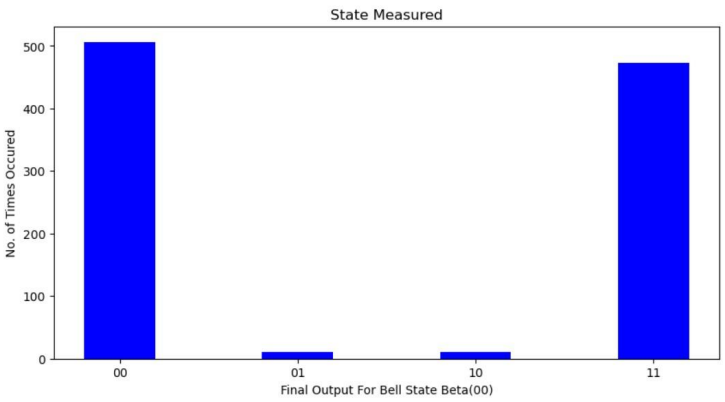
Another major drawback that the platform has is the inability to support complex quantum simulations. The Quantum Processing Unit in Braket is not the best available in the market. The 11 qubits IonQ QPU and the 31-qubit Rigetti Aspen-8 are both too small to run algorithms for quantum computers that are required to exhibit functional quantum supremacy.

The current quantum hardware available in Braket is all remote from the AWS data centers and not co-located. The third-party quantum hardware providers host the Quantum Processing Units. Hence the users' circuits or problems and associated metadata are sent to and processed by the hardware providers. This raises a security concern in terms of user data and circuit design. There will also be a significant increase in latency.

The efficiency of a quantum processor can be calculated using a metric called Quantum Volume. It takes the number of qubits it can handle and the error rates into consideration. Based on this metric, the processors of IBM and Honeywell are better performing than the Amazon Braket processors.

Although Braket may appear to provide a standard interface for all possible devices, the devices themselves have a significant variation to work uniformly.

Though quantum computers with sufficiently uncorrelated and weak noise can solve complex problems, noise should be minimal for better performance and high precision in highly complex problems. The figure shows that the processors (IonQ) provided on Amazon Braket show noise for even a simple Bell state circuit.



# Amazon Braket - Enhancements

Currently, all services on Amazon Braket are under a payment scheme which might be driving a portion of their total user segment away. The payment scheme could hence be restructured, incorporating a free trial for a particular duration or free versions with capacity limits. This would attract new users who are just entering the field and trying to find their ground but do not have the funds to invest immediately. A lot more people will be exposed to the technology and its use cases, increasing the service's popularity. AWS could use its reputation and vast customer base to take giant leaps in the quantum computing space, such as spreading awareness, attracting investors, and growing the field overall.

Phases of improvements will be required to combat the drawbacks that Braket is burdened with. For instance, there is a need for significant enhancements in noise reduction and coherence time which is the amount of time quantum information can be stored, two characteristics that Braket is infamous for. Furthermore, the current Quantum Processing Units provided by Amazon Braket work with a minimal number of qubits, limiting the level of complexity that the processors can deal with. By increasing this number, the simulator can expand the maximum level of the computational complexity of the algorithms that it can handle and could solve many more problems that it cannot solve today.

Besides the available three quantum computers, adding more powerful Quantum Computing services could also attract a wider audience. In general, customers prefer to have choices, and when it comes to a field that is still in its infancy, having options would be beneficial in the long run. Continuous developments coming from multiple sources can be helpful for all the players in the field, which would ultimately develop the quantum computing sphere even further.

In the background of all this, Amazon should also be trying to build their own Quantum Computer. As the field develops, there would be a significant increase in the amount of data involved in the computation, and sharing the data with multiple institutions would cause security concerns. It would also come at a higher cost for Amazon. Having an in-house computer could streamline the process by reducing latency, increasing security, reducing costs, and integrating it with other Amazon products with ease.

# In a nutshell...

Amazon's economic significance in today's world is bound to attract more attention towards quantum computing. It helps bring in the tech giants and the funding required for more research and development into the space. Another notable benefit of AWS is its ubiquity and the customer's familiarity with their permissions and billing systems. It would also provide the user with the ease of integration with other Amazon products. Amazon can take any new technology, invest, and play an integral role in developing quantum computation. They can easily provide access to such services to a large number of people.

While Braket has a quite promising future in the field of quantum computing, it comes with its demerits. D-Wave's Leap and Rigetti's QCS are already functional quantum clouds to which users have access. Hence here, one could say that Braket becomes just another extra layer to user access. If we consider small quantum startups with their own house clouds, Amazon Braket could cause them to fail since users would prefer Braket. Furthermore, with the current processing powers of Amazon Braket, we may not be able to compute powerful quantum algorithms that help us make a difference.

While there may be a few holes in the academic and research spectrum concerning Amazon Braket, it definitely provides a well-defined path for further exploration into the quantum computing space, spreading awareness about the phenomena, and bringing immense talent and capital into the field to move forward in our evergreen quest to learn, explore and push the limits of technology.

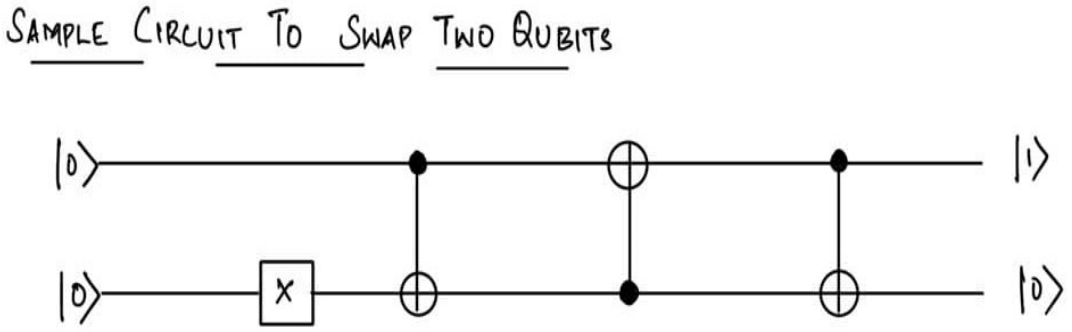
# Coding Assignment

As we've mentioned earlier, a major disadvantage of the Amazon Braket is the lack of a free trial or a free subscription model. Due to this exact reason, the amount of research and development work done using Braket is worryingly limited. As a team, we found it significantly challenging to understand how to install packages, create instances and run them on AWS servers, and implement even the most basic operations and applications we were familiar with. So to tackle this issue, we decided to design and implement various utilities and operations using Amazon Braket and document the process thoroughly. We started with the basic introductory steps such as importing libraries, connecting to the AWS server, setting up a testing circuit, running and viewing the results, etc. We then moved on to designing the universal NAND gate using Toffoli gates, which could easily design any logic circuit of our choice. We implemented half adders, full adders using the above-mentioned NAND gate implementation. For algorithms that perform better than existing classical algorithms, we implemented some algorithms such as superdense coding, Grover's search and showed some of its applications that include searching for a state in a system, to solve various graph colouring problems as well as to use this algorithm to solve different satisfiability constraint problems.

- ## Sample Circuits

In this introductory project file, we aimed to document the building blocks for quantum computing using the Amazon Braket. We've covered how to import libraries, connect to our AWS ID, start working on a device or a simulator using the Braket, and store the results of the computation. We also looked into how to design and implement quantum circuits using basic quantum gates, run those circuits on the device and calculate the results. Additionally, we had also shown how to plot the results that we had calculated.

**Circuit :**

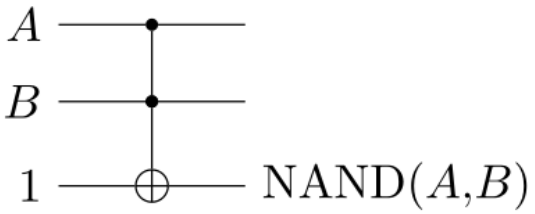


- # Simulating Classical Gates Using Toffoli

Using the simple quantum Toffoli gate, we designed a universal NAND gate such that when the target bit of the Toffoli gate is set to 1, it behaves like a NAND gate. Using this implementation, we further defined basic AND and OR logic gates. The implementation of these basic gates opens a wide range of possibilities as it would enable us to design and implement virtually any logic circuit using the Braket.

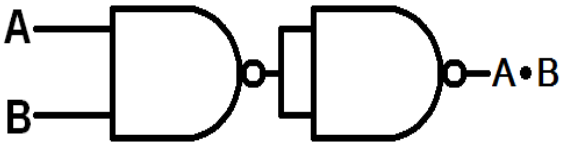
1. **NAND using Toffoli Gate**

**Circuit diagram**




2. **AND Gate**

**Circuit diagram**

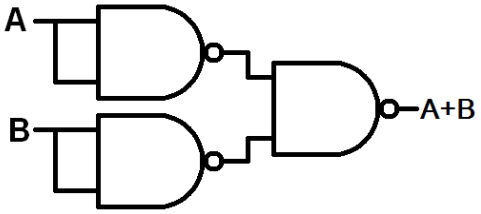



**Code Output**

```
Enter the first number: 10110
Enter the second number: 11100
10110 AND 11100 = 10100
```

3. **OR Gate**

**Circuit diagram**




**Code Output**

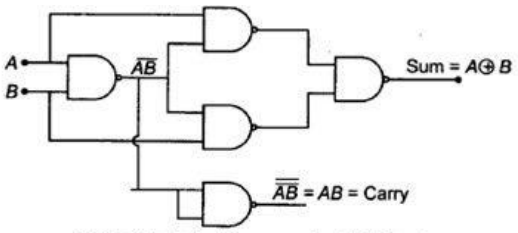
```
Enter the first number: 10110
Enter the second number: 11100
10110 OR 11100 = 11110
```

- # Half Adder and Full Adder

After having implemented the universal NAND gate, we wanted to go a couple of steps further to show how we can use the NAND gate to build other circuits. The circuits that we chose for this were the Half-Adder and the Full-Adder circuits. Using a combination of 5 NAND gates, we implemented the Half-Adder, and then a Full-Adder was then implemented using a combination of 9 NAND gates, which also works for single-bit, as well as n-bit inputs.

1. **Half Adder**

**Circuit Diagram**



Half adder logic diagram using NAND gate

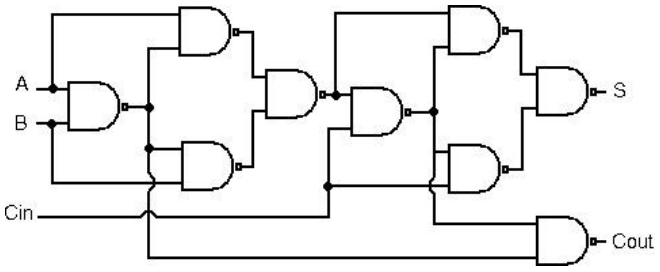
**Code Output**

```
Enter the first number: 1
Enter the second number: 0
Sum = 1 and Carry = 0 when 1 and 0 is added using Half Adder
```

2. **Full Adder**

***For 1 bit***

**Circuit Diagram**




**Code Output**

```
Enter the first number: 1
Enter the second number: 1
Enter the input carry: 1
Sum = 1 and Carry = 1 when 1 and 1 is added using Full Adder with input carry 1
```

***For n bits***

***Binary adder - Full adder is a simple 1–bit adder. If we want to perform n–bit addition, we need an 'n' number of 1–bit full adders that should be used in the form of a cascade connection.***

```
Enter the first number: 12
Enter the second number: 13
12 + 13 = 25
```

# Superdense Coding

Superdense coding is a procedure that allows a person to send two classical bits to another party by just using one qubit. This is done using the property of quantum entanglement. Consider a case where there are two people, say Alice and Bob. In a usual scenario, Alice would need two bits to send 2 bits of data through classical channels. But Alice and Bob could use the quantum properties to use just one qubit to transmit this data. For this, initially, they need to share a maximally entangled pair of qubits(Bell pair). Alice then selects one of the four possible messages that could be sent using two bits (00, 01, 10, 11). Once she selects the messages, she encodes them using the corresponding quantum gates. After this, Alice sends her qubit to Bob, and then he decodes the message by undoing the entangling operation.

**Steps :**

1. Alice and Bob initially share a Bell pair. This can be done by initially starting with two qubits, in this case, two |0> state, and then applying the Hadamard gate (H gate) to the first qubit. Then they are passed through a CNOT gate (CX gate), which produces a bell pair. Alice holds one of the qubits, and Bob has the other qubit.

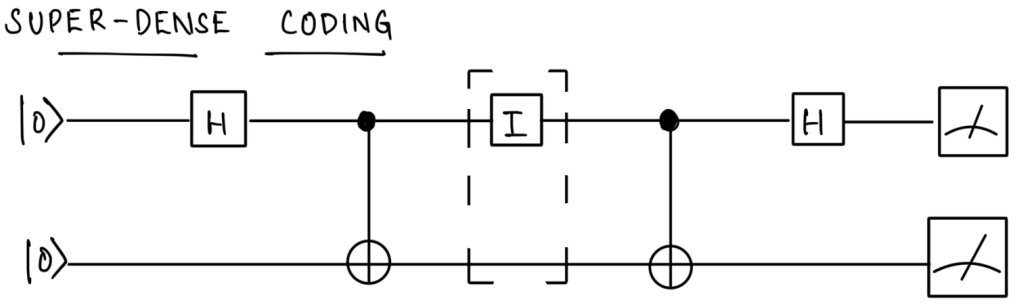
2. Alice then selects the message she wants to send and applies the corresponding quantum gate transformation according to table 1.

3. Alice sends her qubit to Bob. Bob then undoes the entangling by first passing the qubits through a CNOT gate and applying an H gate. This restores the classical message.

| Message | Alice's encoding gate | State Bob receives | After CNOT gate | After $H$ gate (Final) |
|---|---|---|---|---|
| 00 | $I$ | $\|00\rangle + \|11\rangle$ | $\|00\rangle + \|10\rangle$ | $\|00\rangle$ |
| 01 | $X$ | $\|10\rangle + \|01\rangle$ | $\|11\rangle + \|01\rangle$ | $\|01\rangle$ |
| 10 | $Z$ | $\|00\rangle - \|11\rangle$ | $\|00\rangle - \|10\rangle$ | $\|10\rangle$ |
| 11 | $ZX$ | $\|01\rangle - \|10\rangle$ | $\|01\rangle - \|11\rangle$ | $\|11\rangle$ |

***Table 1***

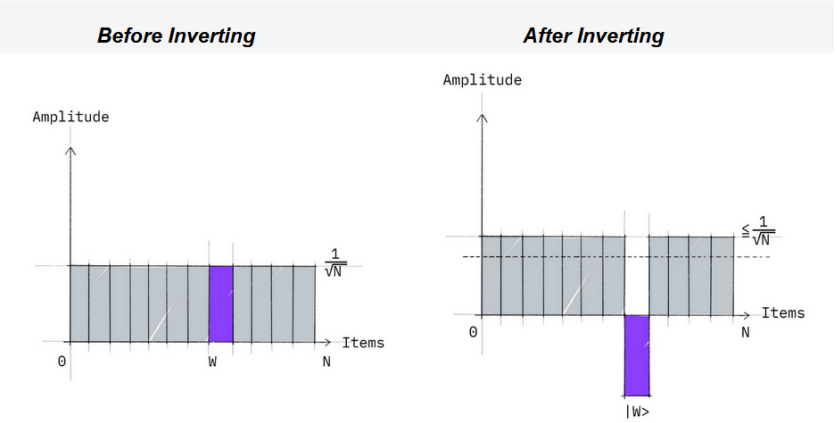
**Circuit Diagram (If Alice Wants To Send 00) :**



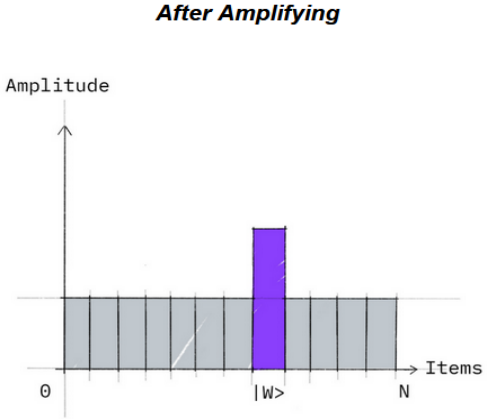
- # Grover's Algorithm

Grover's algorithm is a quantum search algorithm that can be used to search through an unstructured space. It can provide a quadratic speedup over the best classical algorithms. The popularity of Grover's algorithm is due to the importance searching has had over the last century to resolve various issues. Some of Grover's algorithm applications include that they can provide statistical information such as minimum and mean from an unordered dataset much faster than a classical algorithm. It can be used to speed up several encryption problems also.
The Components Used As Part Of Grover's Algorithm:

- **The Oracle:** The oracle function varies depending on the problem. From the output that we receive after applying the Hadamard gate, we see that each state has the same probability of being measured. Once they provide the state we want to search for, the Oracle function will invert the amplitude (multiply with -1) of the state we are searching for.




- **The Amplifier:** Once we invert the amplitude of the target state, we then amplify this state using the amplification operator. We do this on measurement, this state has the most probability of getting selected. Hence, the more we apply both operators, the closer we move towards the target and the higher the probability of a correct answer. We increase its amplitude and, at the same time, shrink the amplitude of all other states.



Through experiments, it will be found that doing this algorithm O(sqrt(n)) times is enough to obtain the correct answer. Hence it provides a quadratic speedup over the best known classical algorithm to search in an unstructured space which takes O(n) time.

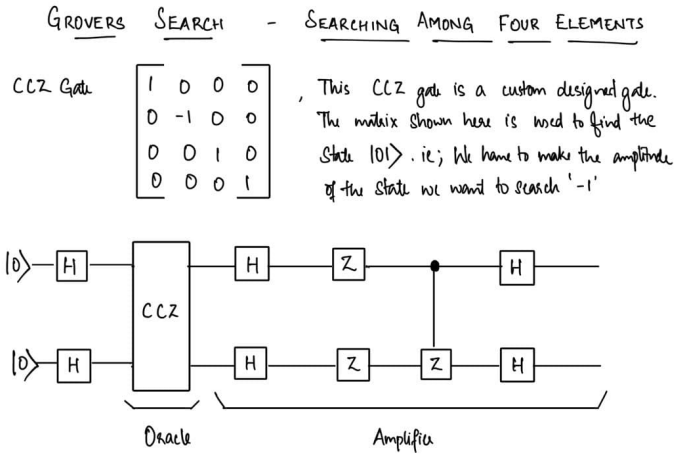
We started by implementing Grover's algorithm on a search space of 4 elements for our initial work. Once we successfully implemented this on Amazon Braket, we expanded our search space to 8 elements. We continued testing the system's capabilities by then testing in on a search space of 16 elements where we prepared 1000 qubits identically and tested the number of times the searched space was returned.

## ○ Application of Grover's Algorithm

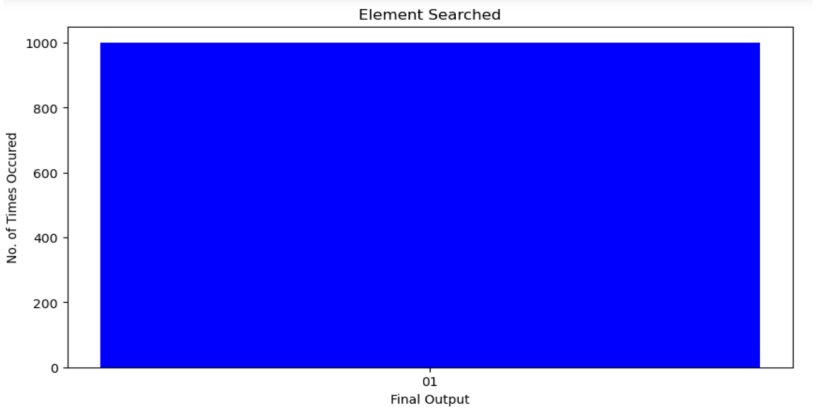
- **Searching Among 4 Elements**

  Here we have displayed an application of Grover's algorithm. We are trying to get the search item from the state space of 2-bit elements ( 00, 01, 10, 11).

**Circuit :**




**Result:**

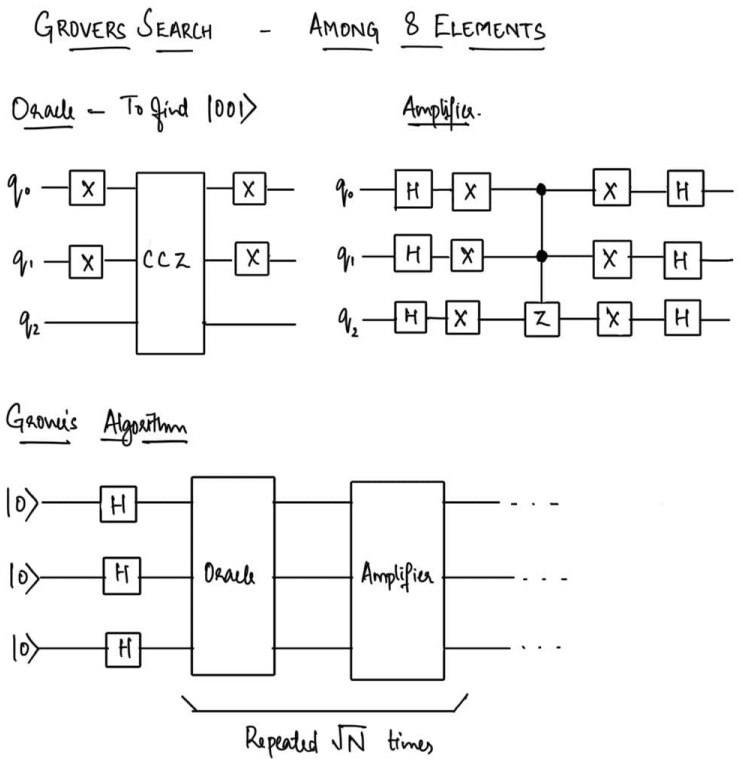



Hence from the above graph, we can conclude that the state with higher number of occurrences **(01)** is the same as the state that we searched for.

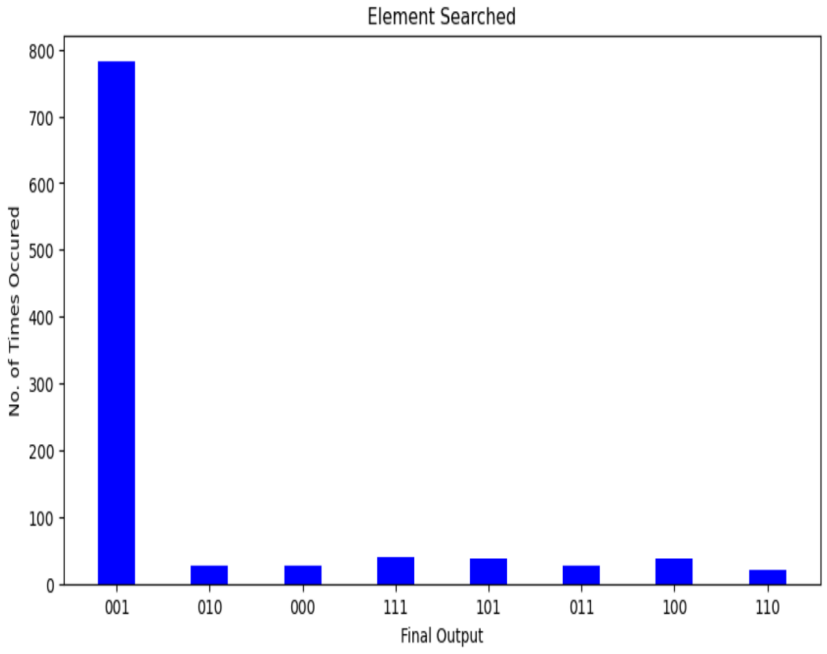
- **Searching Among 8 Elements**

  Searching among 8 elements is an extension of the searching among 4 elements that we previously saw. Here we have created a custom **CCZ gate** that is used for inverting. As for any Grover's algorithm here it is then passed through an Oracle and then through an Amplifier. The below diagram can be referred to get a clear understanding of how the Circuit is created.

**Circuit :**




**Result :**

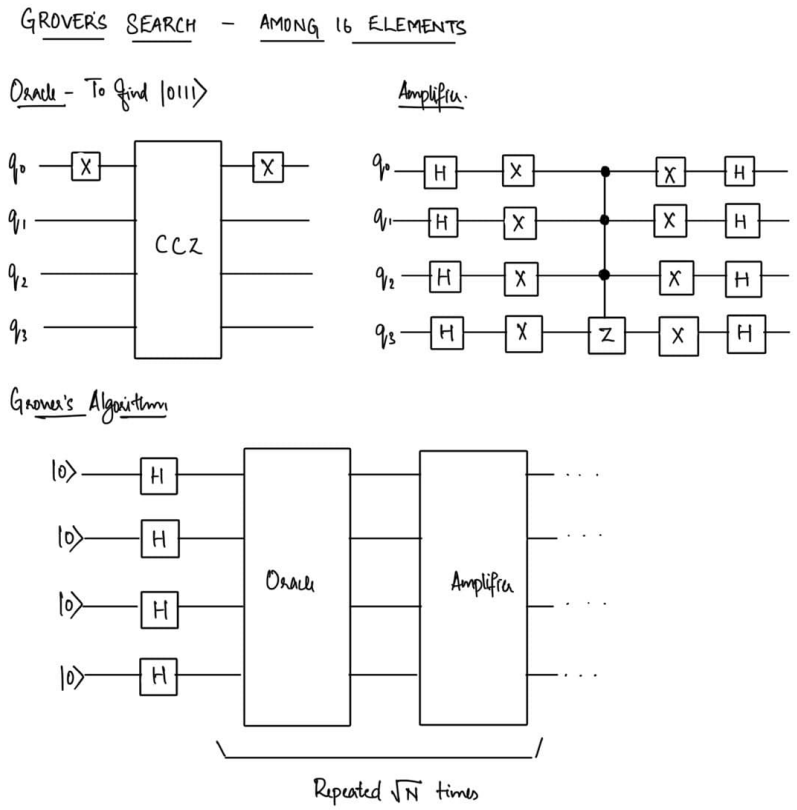



Hence from the above graph we can conclude that the state with higher number of occurrences **(001)** is the same as the state that we searched for.

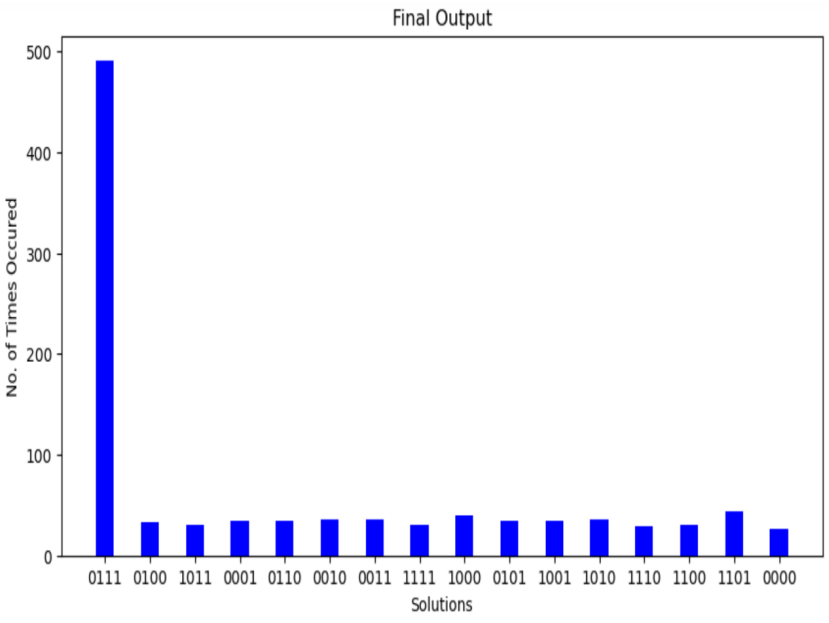
- **Searching Among 16 Elements**

Searching among 16 elements is an extension of the searching among 4 and 8 elements that we previously saw. Similar to the previous case we have created a custom Cz gate that is used for inverting. As for any Grover's algorithm here it is then passed through an Oracle and then through an Amplifier. The below diagram can be referred to get a clear understanding of how the Circuit is created.

**Circuit :**




**Result :**




Hence from the above graph, we can conclude that the state with a higher number of occurrences **(0111)** is the same as the state that we searched for, and therefore, our search was successful.

- **Graph Colouring Using Grover's Algorithm - 4 Vertices Problem**

The graph colouring problem that we consider requires the condition that no two adjacent nodes of the matrix (horizontally and vertically adjacent) can have the same colour (denoted by 0 and 1, representing two colours). Hence, we need to modify our Oracle to consider these constraints. We can consider the below example -
If the indices of the matrix are as follows :

| | |
|---|---|
| 0 | 1 |
| 2 | 3 |

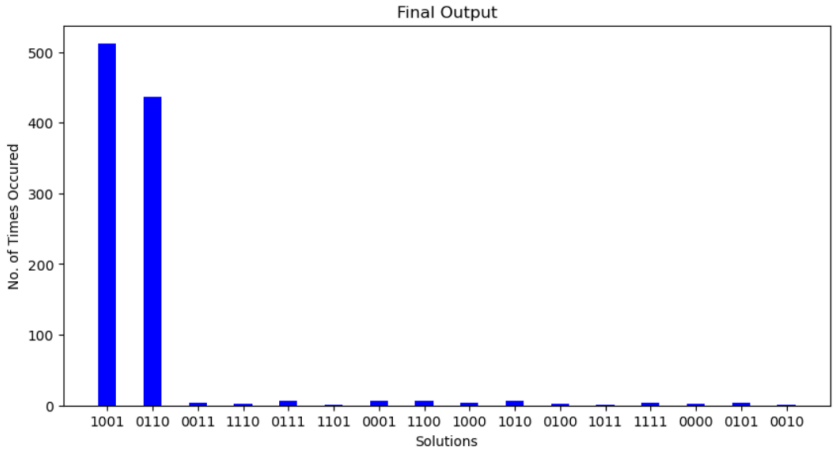
Then the following list of pairs of indices cannot be coloured the same.

**[ [0,1], [0,2], [1,3], [2,3] ]**

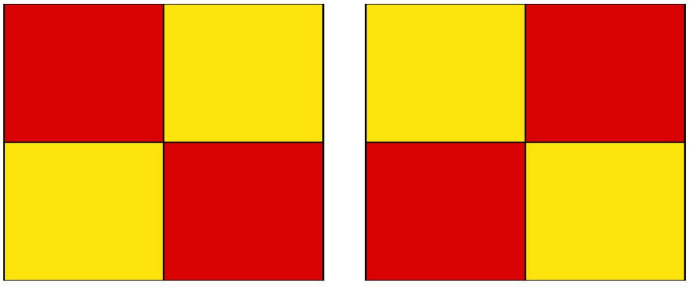
Hence, the program can either colour it as

- Solution 1:
  - Vertex 0 and 3 - Colour A, Vertex 1 and 2 - Colour B
- Solution 2:
  - Vertex 0 and 3 - Colour B, Vertex 1 and 2 - Colour A

**Result:**




The resulting graph shows that solutions **0110** and **1001** are the overwhelming majority, consistent with our required solution set.



These colour patterns are possible solutions to colour the graph if we consider 0 to be Red and 1 to be Yellow.

- **Graph Colouring Using Grover's Algorithm - 5 Vertices problem**

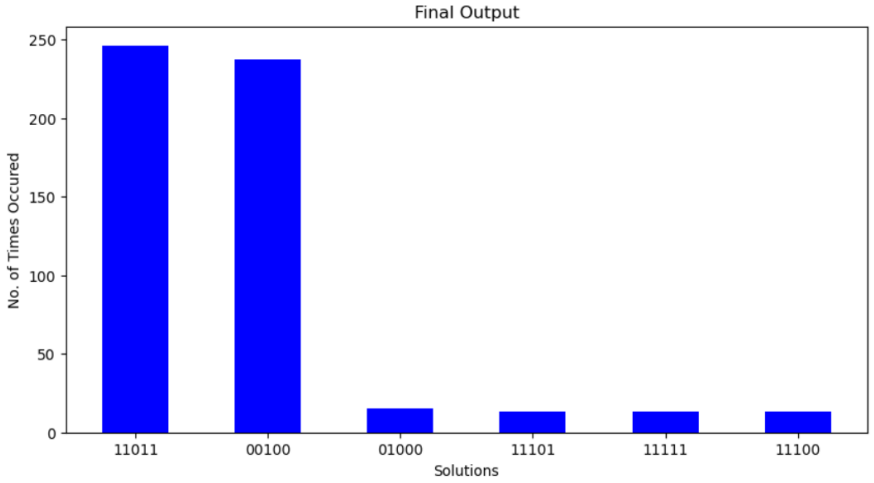
Now that we could solve a 2x2 graph colouring problem, we decided to expand the problem to 5 qubits and increase the constraints to see how the algorithm performs.

For 5 qubits, the graph that we decided to go with was similar to that of a cross i.e One qubit in the intersection of the cross, and four qubits in the four edges of the cross.

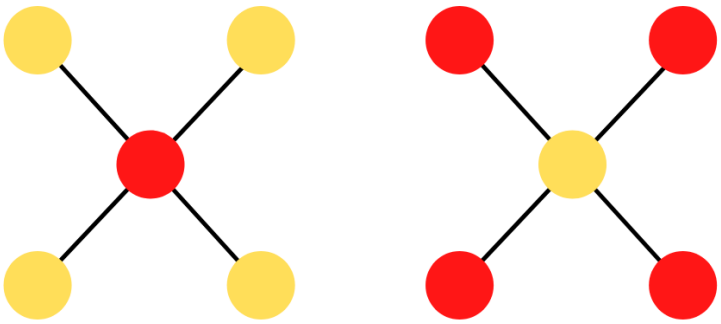
Applying the constraints that no two adjacent nodes can have the same colour, and that the minimum number of colours must be used to colour the graph, we can devise the constraints that the centre node (intersection of the cross) must be of one colour (say 0), and the others must all have the other colour (say 1). So with 5 qubits the output must be -

# **00100** OR **11011**

**Result:**




From the resulting graph, we can see that solutions **11011 and 00100** have occurred the most number of times significantly, consistent with our solution set.



These colour patterns are possible solutions to colour the graph if we consider 0 to be Red and 1 to be Yellow.

# ● Satisfiability Problem

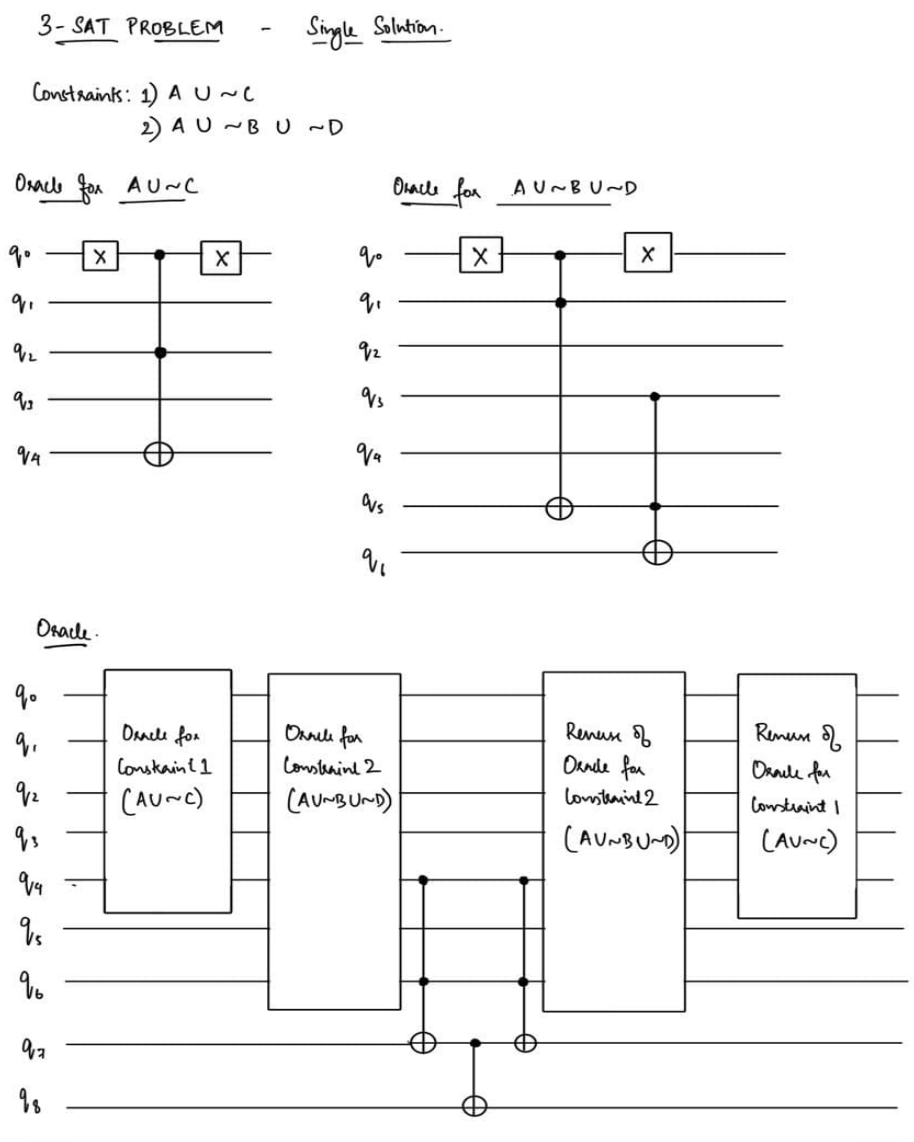
The Boolean satisfiability problem is a problem wherein we need to determine if a given boolean expression is satisfiable, i.e. if given a boolean formula, can we change the variables by TRUE or FALSE so that formula evaluates to TRUE. The boolean expression or clause is basically a combination of variables and operators enclosed within parentheses. The operators used are:

- AND ( denoted by ∧ )
- OR ( denoted by ∨ )
- NOT ( denoted by ¬ )

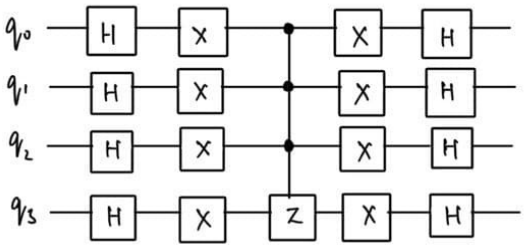
This is in general a constraint satisfaction problem. Below, we show how such constraint satisfaction problems with a single solution can be solved using Grover's algorithm by changing its oracle component.

$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee x_3 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_2} \vee x_4)$$

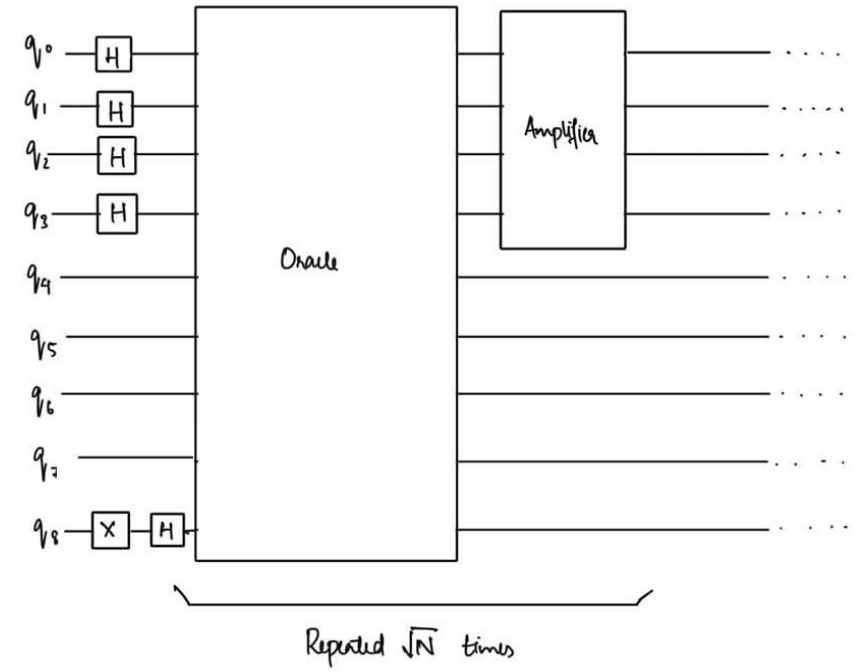
## 1. Satisfiability Problem With A Single Solution



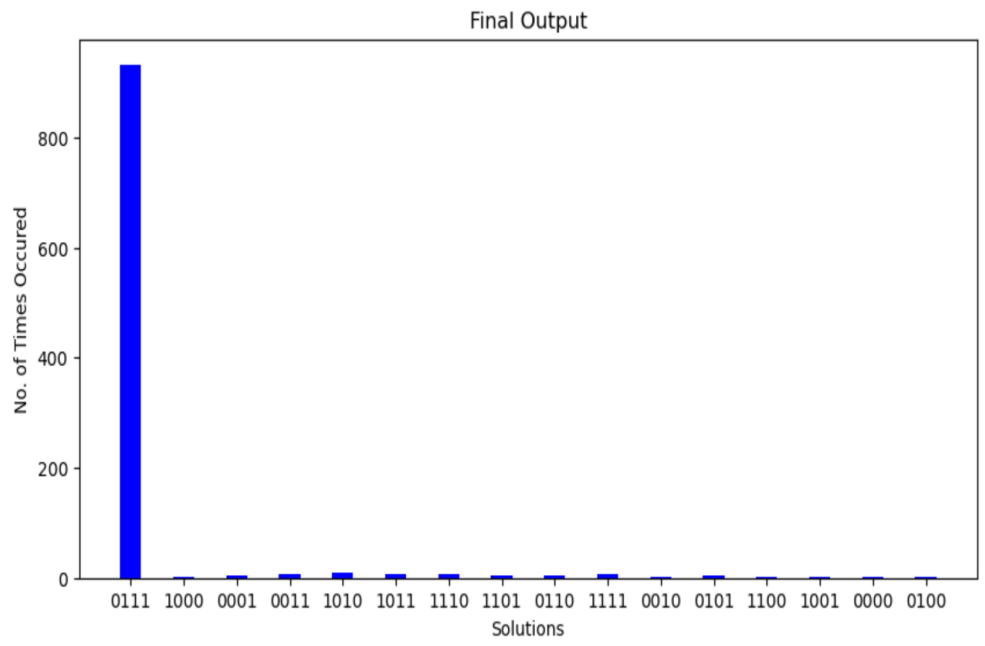
Amplifier




Grover's Algorithm




**Result :**




If we wanted to obtain a 0 for the problem -

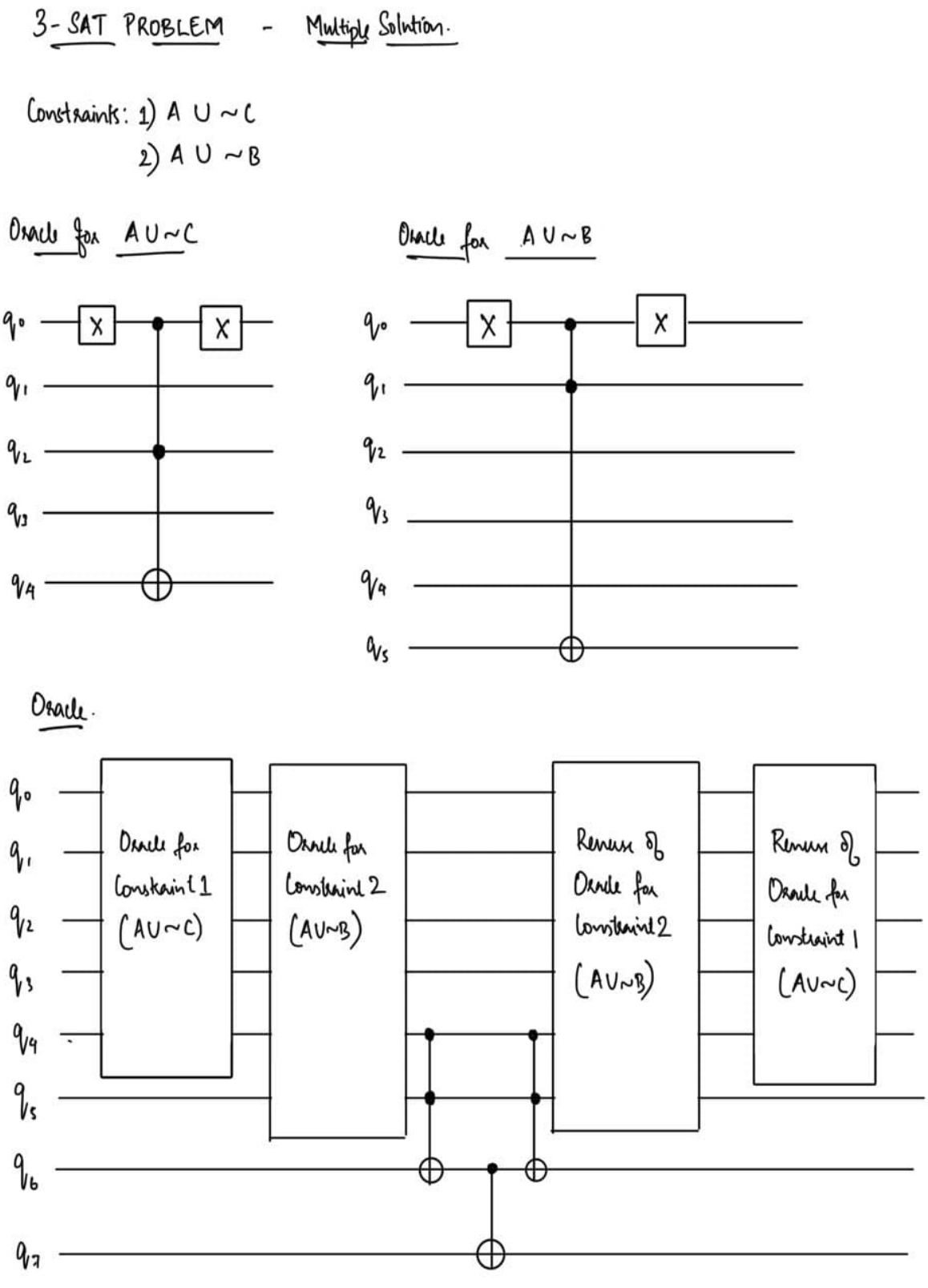
The Solutions are:

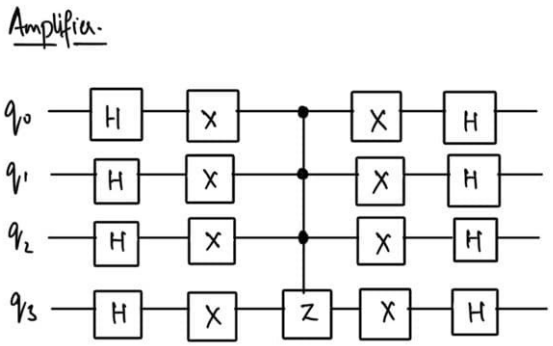
**A-0**
**B-1**
**C-1**
**D-1**

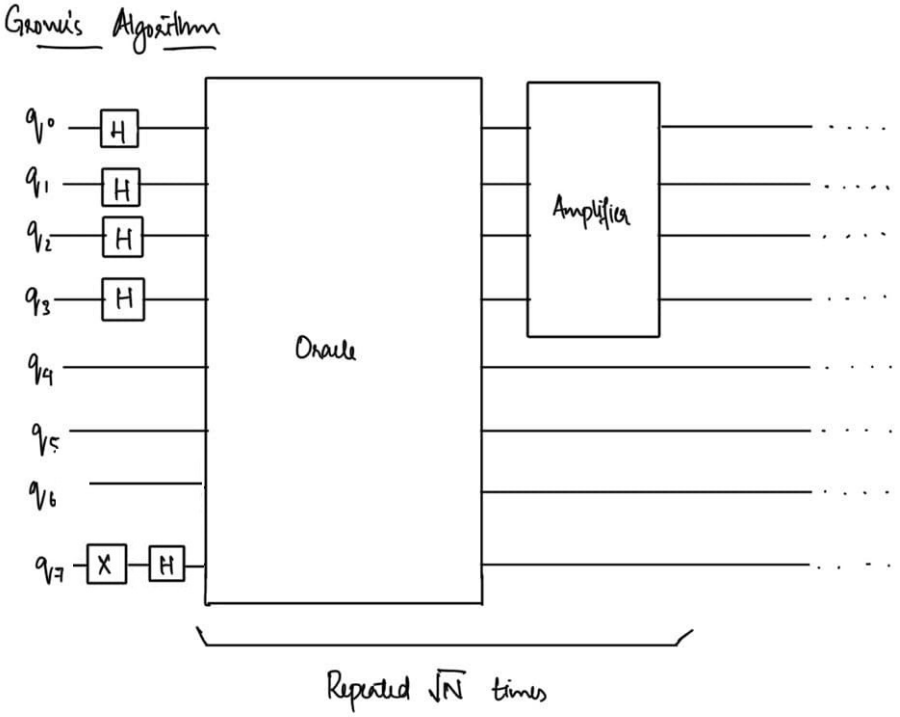
## 2. Satisfiability Problem With Multiple Solution

Below, we show how such constraint satisfaction problems with multiple solutions can be solved using Grover's algorithm by changing its oracle component.

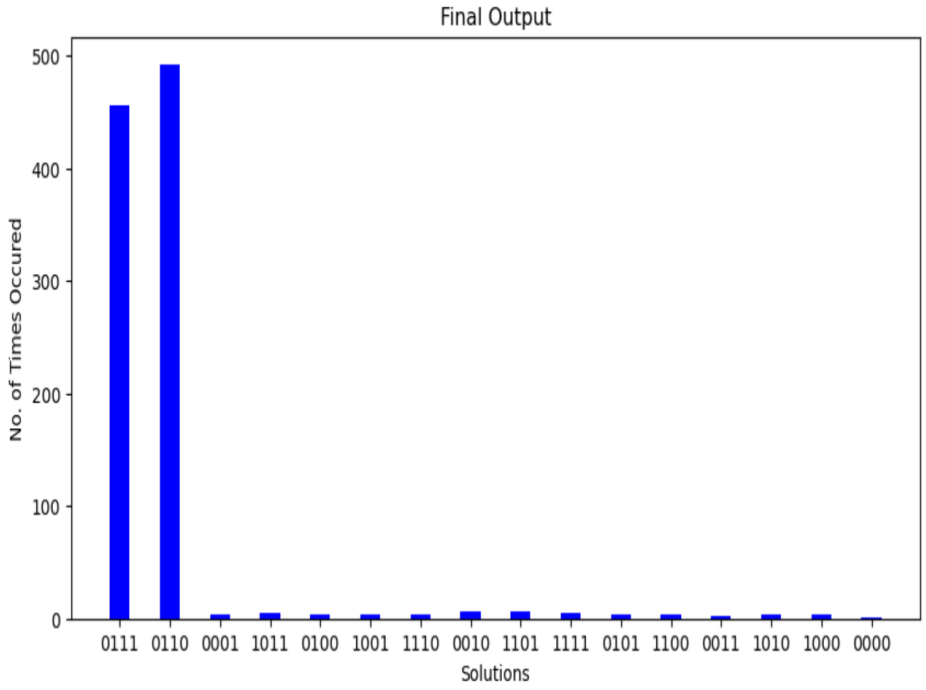
**Circuit :**






**Result :**




If we wanted to obtain a 0 for the problem -

The Possible Solutions are:

**A-0, B-1, C-1, D-1**

**Or**

**A-0, B-1, C-1, D-0**